CLAIMS

1. A method for removing a part of a computing process, wherein said process splits into a first process and a second sub-process, the second sub-process comprising items of data and/or program code and/or execution states of said computing process not required by said first process.

2. A method as claimed in claim 1 wherein said sub-process comprises items of data, program code and execution state of said computing process.

3. A method as claimed in claim 1 or 2 wherein a construct is formed for storing said sub-process, while said first process retains the process identity of said computing process.

4. A method as claimed in claim 3 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a new sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said computing process.

5. A method as claimed in claim 4 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

6. A method as claimed in any of claims 3 to 5 wherein said construct is provided with an authorising signature.

7. A method as claimed in any of claims 3 to 6 wherein said construct is sent to a memory storage device.

8. A method as claimed in any of claims 3 to 7 wherein said sub-process is a dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and wherein said first process is able to re-acquire data, program codes and execution states from said dormant process as and when required by said first process.

9. A method as claimed in claim 8 wherein said first process re-acquires data, program code and execution states from said dormant process falling within specified ranges.

10. A method as claimed in claim 8 or 9 wherein said sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said second process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

11. A method as claimed in any of claims 3 to 10 wherein after said first process finishes executing data, program modules and execution states from said first process are added to said sub-process and said sub-process is reactivated.

12. A method as claimed in claim 11 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said sub-process prior to adding said first process to said sub-process.

13. A method as claimed in claim 1 or 2 wherein a construct is formed for storing said sub-process, while said first process is run in place of said computing process.

14. A method as claimed in claim 13 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said first process.

15. A method as claimed in claim 14 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

16. A method as claimed in any of claims 13 to 15 wherein said construct is provided with an authorising signature.

17. A method as claimed in any of claims 13 to 16 wherein said construct is sent to a memory storage device.

18. A method as claimed in any of claims 13 to 17 wherein said sub-process is a dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and wherein said first process is able to re-acquire data, program codes and execution states from said dormant process as and when required by said first process.

19. A method as claimed in claim 18 wherein said first process re-acquires data, program code and execution states from said dormant process falling within specified ranges.

20. A method as claimed in claim 18 or 19 wherein said sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said sub-process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

21. A method as claimed in any of claims 13 to 20 wherein after said first process finishes executing data, program modules and execution states from said first process are added to said sub-process and said sub-process is reactivated.

22. A method as claimed in claim 21 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said sub-process prior to adding said first process to said sub-process.